Introduction to APIs in JavaScript

APIs (Application Programming Interfaces) are essential tools that allow different programs or services to communicate with one another. In this module, we explored the foundations of APIs and how to interact with them using JavaScript.

Why Do We Need APIs?

Access Real Data: APIs let us retrieve live, dynamic data from external sources.

Build More Interesting Apps: Enabling features like weather info, random jokes, or social media integration.

Prepare to Learn Full Stack: APIs are the glue between the frontend and backend in full-stack development.

☑ What We Learned in This Module

- Basics and BoredBot

- URLs, REST, and BlogSpace

- Async JavaScript

What is an API?

• A tool that allows your program to interact with another program.

• You don't get full access—only what the other service wants to expose.

Restaurant Analogy: You (client) make an order (request) through a waiter (API), and the kitchen (server) sends back your food (response).

Examples of APIs

1. Getting Data From a Server

  • Servers host APIs with specific endpoints.

  • We make requests to these endpoints to get only the allowed data.

2. Pre-written Code APIs

  • DOM API: document.getElementById()

  • Array Methods API: .map(), .filter(), etc.

  • 3rd Party Packages: External libraries offering predefined functionalities.

Clients and Servers

- Client: A device or browser that makes requests to the server.
- Server: Accepts requests and returns responses (data, HTML, etc.).
- Note: Servers can sometimes act as clients too, requesting data from other servers.

The Request and Response Cycle

1. Client sends a request to a specific URL.
2. Server processes the request and sends back a response.
3. Client receives and uses the data (typically in JSON format).

JSON (JavaScript Object Notation)

- A lightweight data format used to share data between client and server.
- Similar to JavaScript objects.
- Use https://jsonlint.com/ to validate JSON format.

Viewing API Calls in the Browser

- Go to Developer Tools → Network Tab → XHR to monitor requests.
- You can inspect request URL, status code, and response data.

🧪 Basic Fetch API Example

```
console.log("The first console log")

fetch("https://dog.ceo/api/breeds/image/random")
    .then(response => response.json())
    .then(data => console.log(data))

console.log("The second console log")

for (let i = 0; i < 100; i++) {
    console.log("I'm inside the for loop")
}
```

JavaScript does not wait for the fetch to complete. It continues executing the code. This behavior is due to JavaScript being asynchronous and non-blocking.

🎉 Real-world Example: BoredBot

```
fetch("https://apis.scrimba.com/bored/api/activity")
    .then(response => response.json())
    .then(data => {
        console.log(data)
        document.getElementById("activity-name").textContent = data.activity
    })
```

- Makes a call to the Bored API.
- Updates the content on the page dynamically using the fetched data.

Resources

- JSONLint – Validate JSON: https://jsonlint.com/

- Dog API: https://dog.ceo/dog-api/

- Bored API: https://www.boredapi.com/